REMARKS

Claims 1-25 are all the claims presently pending in the application. Claims 1, 10, 17-18 and 20 have been amended to more particularly define the invention.

It is noted that the claim amendments are made only for more particularly pointing out the invention, and not for distinguishing the invention over the prior art, narrowing the claims or for any statutory requirements of patentability. Further, Applicant specifically states that no amendment to any claim herein should be construed as a disclaimer of any interest in or right to an equivalent of any element or feature of the amended claim.

Claims 21-22 stand rejected under 35 U.S.C. §112, first paragraph as allegedly not enabled by the specification.

Applicant notes that claims 21-22 are not subject to any prior art rejection and would, therefore, be presumably allowable except for the alleged informalities.

Claims 1-3, 5-20 and 23 stand rejected under 35 U.S.C. §103(a) as allegedly being unpatentable over Peterson et al. (U. S. Patent No. 6,593,940) in view of Flanagan et al. (U. S. Patent No. 6,343,371).

Applicant notes that the Examiner incorrectly indicates on the Summary page and on page 1 of the Office Action that only claims 1-23 are pending. The Examiner does not expressly indicate that claims 24 and 25 are subject to any rejections and therefore, these claims are presumably allowable.

This rejection is respectfully traversed in the following discussion.

I. THE CLAIMED INVENTION

The claimed invention (e.g., as recited in claim 1) is directed to a method for statically detecting a datarace condition in a multithreaded application. The method includes inputting a set of input information including a multithreaded context graph, processing the set of input information by comparing threads that may execute statements in a statement pair, and outputting a statement conflict set that identifies the statement pairs whose execution instances definitely or potentially cause dataraces, such that a datarace condition is statically detected without executing the multithreaded application.

Importantly, the processing includes computing a node conflict set by traversing the multithreaded context graph to identify conflicting pairs of nodes in the multithreaded context graph (Application at page 16, lines 3-17; Figure 4).

Conventional datarace detection methods identify many dataraces that may never be exhibited in any program execution (e.g., "false positives"). In addition, programmer annotations are required to improve the effectiveness of such conventional tools (Application at page 6, lines 1-10).

The claimed invention, on the other hand, computes a node conflict set by traversing the multithreaded context graph to identify conflicting pairs of nodes in the multithreaded context graph (Application at page 16, lines 3-17; Figure 4). This feature helps to allow the claimed method (e.g., and system) to more accurately detect a datarace condition (e.g., a condition where a datarace definitely will occur or may possibly occur) than in conventional methods (Application at page 6, lines 12-15).

II. THE 35 USC 112, SECOND PARAGRAPH REJECTION

The Examiner alleges that claims 21-22 are not enabled by the specification. Applicant again submits, however, that these claims are clearly enabled by the specification.

Applicant would again point out that one of ordinary skill in the art would readily understand that the phrase "tagging a statement with a set of threads" **is a non-mathematical explanation of the following notations referred to on pages 19 and 21 in the present Application**:

MustThreadObj(p(S.i)) and MayThreadObj(p(S.i)),

where the former means the set of threads, tagged for statement p, that ***must*** execute p, and the latter means the set of threads, tagged for p, that ***may*** execute p.

With respect to claim 22 which recites "*comparing sets of locks held by threads that may execute said statements*", Applicant would again point out that the term "lock" may refer to synchronization objects which are clearly described in the specification, for example, at page 10, line 13.

Applicant again respectfully submits that practitioners of multithreaded application developers understand well that the two terms are synonymous with each other. Computing the synchronization objects of a statement is described in detail on page 15, lines 10 to 12 of the specification. Further, "Comparing sets of locks" is clearly described in the specification, for example, on page 19, line 24-page 20, line 8, which states:

> *"The synchronization objects of two statements allow the determination of whether two statements (or two executions of a single statement) executed by two different threads are synchronized with respect to each other via the synchronization objects (e.g., of synchronized methods and synchronized blocks)."*

Thus, Applicant respectfully submits that claims 21 and 22 are clearly enabled by the specification.

In view of the foregoing, the Examiner is respectfully requested to withdraw this rejection.

III. THE ALLEGED PRIOR ART REFERENCES

The Examiner alleges that Peterson would have been combined with Flanagan to form the invention of claims 1-20 and 23. Applicant submits, however, that these alleged references would not have been combined and even if combined, the alleged combination would not teach or suggested each and every feature of the claimed invention.

Peterson discloses a method of finding errors in multithreaded applications. Races are instanced during the execution of the program. (Peterson at Abstract).

Flanagan discloses a system for statically detecting potential race conditions in a multi-threaded computer program. In the system, a race condition detector determines whether accesses to object data fields are consistently protected by an appropriate lock. An object data field access that is not protected by an appropriate lock indicates a potential race condition (Flanagan at Abstract).

Applicant respectfully submits that these references are completely unrelated, and no person of ordinary skill in the art would have considered combining these disparate references, absent impermissible hindsight.

In fact, Applicant submits that the references provide no motivation or suggestion to urge the combination as alleged by the Examiner. Indeed, these references clearly do not teach or suggest their combination. Therefore, Applicant respectfully submits that one of ordinary skill in the art would not have been so motivated to combine the references as alleged by the Examiner. Therefore, the Examiner has failed to make a prima facie case of obviousness.

However, neither Peterson, nor Flanagan, nor any alleged combination thereof teaches or suggests "*processing the set of input information by comparing threads that may execute statements in a statement pair, said processing* ***comprising computing a node conflict set by traversing said multithreaded context graph to identify conflicting pairs of nodes in said multithreaded context graph***", as recited, for example, in claim 1 and similarly recited in claims 17, 20 and 23 (Application at page 16, lines 3-17; Figure 4).

As noted above, this feature helps to allow the claimed method (e.g., and system) to more accurately detect a datarace condition (e.g., a condition where a datarace definitely will occur or may possibly occur) than in conventional methods (Application at page 6, lines 12-15).

Clearly, this feature is not taught or suggested by Peterson. Indeed, the Examiner allege that Peterson discloses a multithreaded context graph at col. 6, lines 1-5. That is, the Examiner surprisingly attempts to equate the "monitor lock cycle graph" in Peterson with a multithreaded context graph of the invention. This is clearly incorrect.

Indeed, Applicant would point out that the multithreaded context graph (MCG) in the claimed invention may include a call graph that additionally represents synchronized blocks and methods as separate nodes (Application at page 14, lines 3-5). The monitor lock cycle graph in Peterson, however, is "a persistent record connecting locks that are held to locks that are trying to be acquired" (Peterson at col. 6, lines 1-5). Thus, the monitor lock cycle graph in Peterson is completely unrelated to the MCG of the claimed invention.

Further, even assuming (arguendo) that the monitor lock cycle graph can somehow be equated with the MCG of the claimed invention, Peterson certainly does not teach or suggest **computing a node conflict set by traversing the multithreaded context graph to identify conflicting pairs of nodes in the multithreaded context graph**. Indeed, nowhere does Peterson teach or suggest traversing the monitor lock cycle graph, let alone **traversing it to identify conflicting pairs of nodes.**

Moreover, nowhere does Peterson teach or suggest the concept of a none conflict set as in the claimed invention.

Therefore, Peterson clearly does not teach or suggest processing the set of input information by computing a node conflict set by traversing said multithreaded context graph to identify conflicting pairs of nodes in said multithreaded context graph.

Likewise, this feature is not taught or suggested by Flanagan. Indeed, Flanagan simply states the properties that need be checked, which are the conditions for a datarace. What matters is **how the properties are checked**, which is an important aspect of the claimed invention.

Applicant would again point out that that is at least one area where Flanagan's invention differs from the claimed invention. Using the analogy of the null-pointer error or the divide-by-zero error above, Flanagan's statement corresponds to saying that null-pointers and divide-by-zero need be checked, which is, needless to say, true for null-pointer or divide-by-zero checking.

In short, nowhere does Flanagan teach or suggest processing the set of input information by **computing a node conflict set by traversing said multithreaded context graph to identify conflicting pairs of nodes in said multithreaded context graph.**

Thus, Flanagan is clearly unrelated to the claimed invention. Thus, Flanagan clearly does not make up for the deficiencies of Peterson.

Therefore, Applicant submits that these alleged references would not have been combined and even if combined, the combination would not teach or suggest each and every element of the claimed invention. Therefore, the Examiner is respectfully requested to withdraw this rejection.

III. FORMAL MATTERS AND CONCLUSION

In view of the foregoing, Applicant submits that claims 1-25, all the claims presently pending in the application, are patentably distinct over the prior art of record and are in condition for allowance. The Examiner is respectfully requested to pass the above application to issue at the earliest possible time.

Should the Examiner find the application to be other than in condition for allowance, the Examiner is requested to contact the undersigned at the local telephone number listed below to discuss any other changes deemed necessary in a telephonic or personal interview.

The Commissioner is hereby authorized to charge any deficiency in fees or to credit any overpayment in fees to Assignee's Deposit Account No. 50-0510.

Respectfully Submitted,

Date: 10/17/06

[illegible]

Phillip E. Miller, Esq.
Registration No. 46,060

McGinn IP Law Group, PLLC
8321 Old Courthouse Road, Suite 200
Vienna, VA 22182-3817
(703) 761-4100
Customer No. 21254

CERTIFICATE OF FACSIMILE TRANSMISSION

I hereby certify that the foregoing Amendment was filed by facsimile with the United States Patent and Trademark Office, Examiner Chuck D. Kendall, Group Art Unit # 2192 at fax number (571) 273-8300 this 17th day of October, 2006.

[illegible]

Phillip E. Miller
Reg. No. 46,060